Threading and Multiprocessing

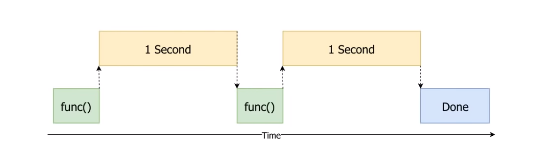
# Introduction

We want to use threading when it is going to significantly speed up our settings.

Basic example running sleep methods concurrently (at the same time).

import time  
  
start = time.perf\_counter*()*def do\_something*()*:  
 print*(*'Sleeping 1 second...'*)* time.sleep*(*1*)* print*(*'Done sleeping.'*)*do\_something*()*do\_something*()*finish = time.perf\_counter*()*print*(*f'Finished in *{*round*(*finish-start, 4*)}* second(s)'*)*

Graph of what happens in the background.



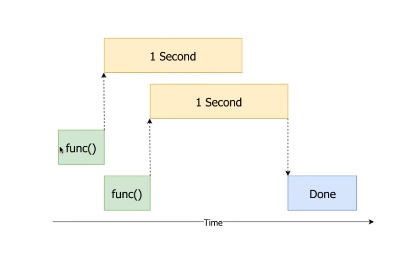
Running everything in order like this is called running in it synchronously .

These are called CPU-bound and IO bound tasks.

CPU-Bound – are crunching a lot of numbers and using the CPU

IO-Bound – Waiting for input and output operations to be completed (reading and writing , downloading stuff etc.)

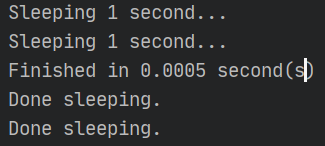
# How our code looks with threading



## Example with threading (with Thread.start())

import time  
import threading  
  
start = time.perf\_counter*()*def do\_something*()*:  
 print*(*'Sleeping 1 second...'*)* time.sleep*(*1*)* print*(*'Done sleeping.'*)*t1 = threading.Thread*(*target=do\_something*)*t2 = threading.Thread*(*target=do\_something*)*t1.start*()*t2.start*()*finish = time.perf\_counter*()*print*(*f'Finished in *{*round*(*finish-start, 4*)}* second(s)'*)*

So what happens here.



So this here runs the individual bits of our functions and we can see that it starts sleeping and immediately after it does that it says that it’s finishes and after 1 second our sleep() function is terminated and tells us it’s done sleeping.

## The join() method

t1.join*()*t2.join*()*

Calling the join() method on each thread makes sure that they complete before moving on to calculating the finishes line.